

# Betriebssysteme (BS)

## VL 8.2 – Koroutinen und Fäden – Fortsetzungsmodell

**Volkmar Sieh / Daniel Lohmann**

Lehrstuhl für Informatik 4  
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität  
Erlangen Nürnberg

WS 21 – 13. Dezember 2021



[https://www4.cs.fau.de/Lehre/WS21/V\\_BS](https://www4.cs.fau.de/Lehre/WS21/V_BS)

# Agenda

---

Motivation

**Grundbegriffe**

Routine und asymmetrisches Fortsetzungsmodell

Koroutine und symmetrisches Fortsetzungsmodell

Implementierung

Ausblick

Zusammenfassung

Referenzen



# Grundbegriffe: Routine, Kontrollfluss

- **Routine:** eine endliche Sequenz von Anweisungen
  - z. B. die Funktion  $f$
  - Sprachmittel fast aller Programmiersprachen
  - wird ausgeführt durch **(Routinen-)Kontrollfluss**
- **(Routinen-)Kontrollfluss:** eine Routine in Ausführung
  - **Ausführung** und **Kontrollfluss** sind synonyme Begriffe
  - z. B. die Ausführung  $\langle f \rangle$  der Funktion  $f$ 
    - beginnt bei Aktivierung mit der ersten Anweisung von  $f$

Zwischen **Routinen** und **Ausführungen** besteht eine **Schema–Instanz Relation**. Zur klaren Unterscheidung werden die Instanzen ( $\mapsto$  Ausführungen) deshalb hier in spitzen Klammern gesetzt:

$\langle f \rangle$ ,  $\langle f' \rangle$ ,  $\langle f'' \rangle$  sind Ausführungen von  $f$ .



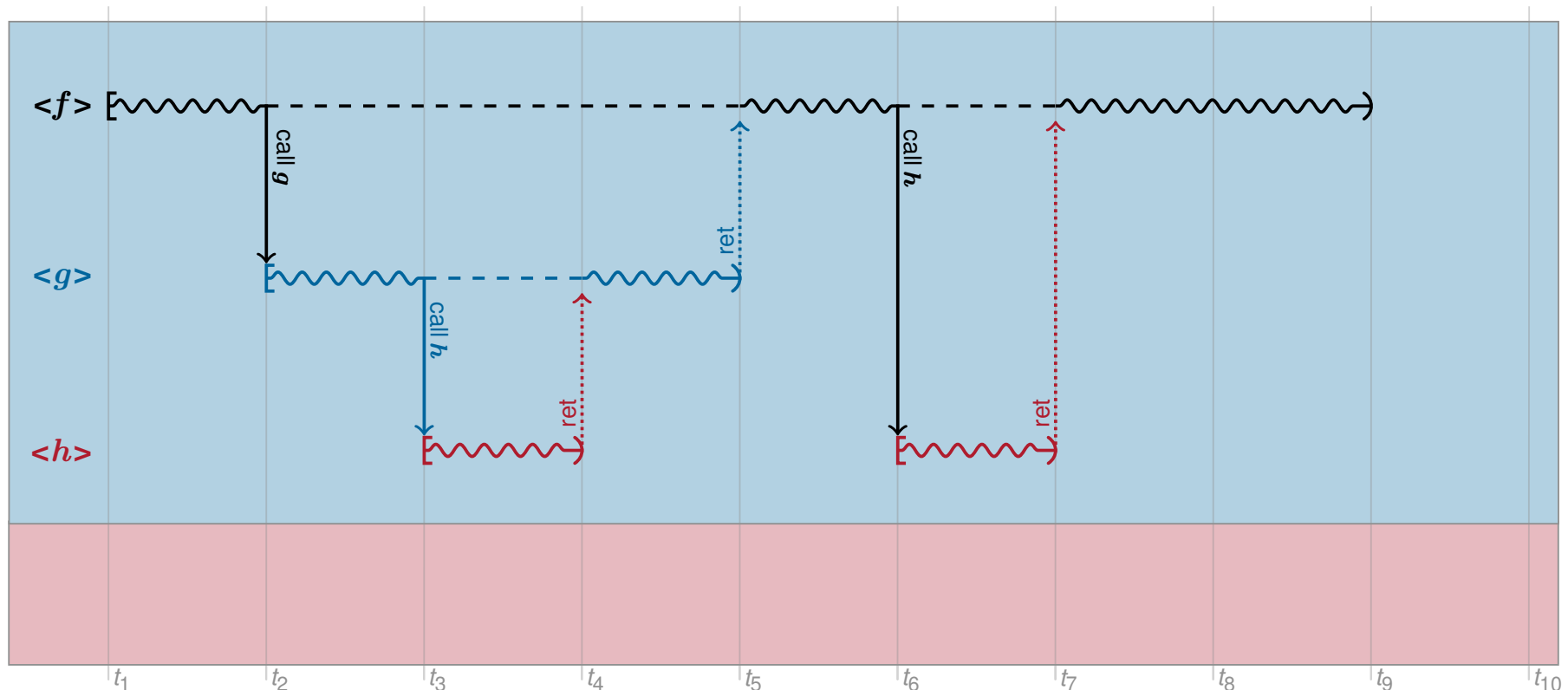
# Grundbegriffe: Routine, Kontrollfluss

- Routinen-Kontrollflüsse werden erzeugt, gesteuert, und zerstört mit speziellen **Elementaroperationen**
  - $\langle f \rangle$  *call*  $g$  (Ausführung  $\langle f \rangle$  erreicht Anweisung `call g`)
    1. **erzeugt** neue Ausführung  $\langle g \rangle$  von  $g$
    2. **suspendiert** die Ausführung  $\langle f \rangle$
    3. **aktiviert** die Ausführung  $\langle g \rangle$   
( $\rightsquigarrow$  erste Anweisung wird ausgeführt)
  - $\langle g \rangle$  *ret* (Ausführung  $\langle g \rangle$  erreicht Anweisung `ret`)
    1. **suspendiert** die Ausführung  $\langle g \rangle$
    2. **zerstört** die Ausführung  $\langle g \rangle$
    3. **reaktiviert** die Ausführung des Vater-Kontrollflusses (z. B.  $\langle f \rangle$ )



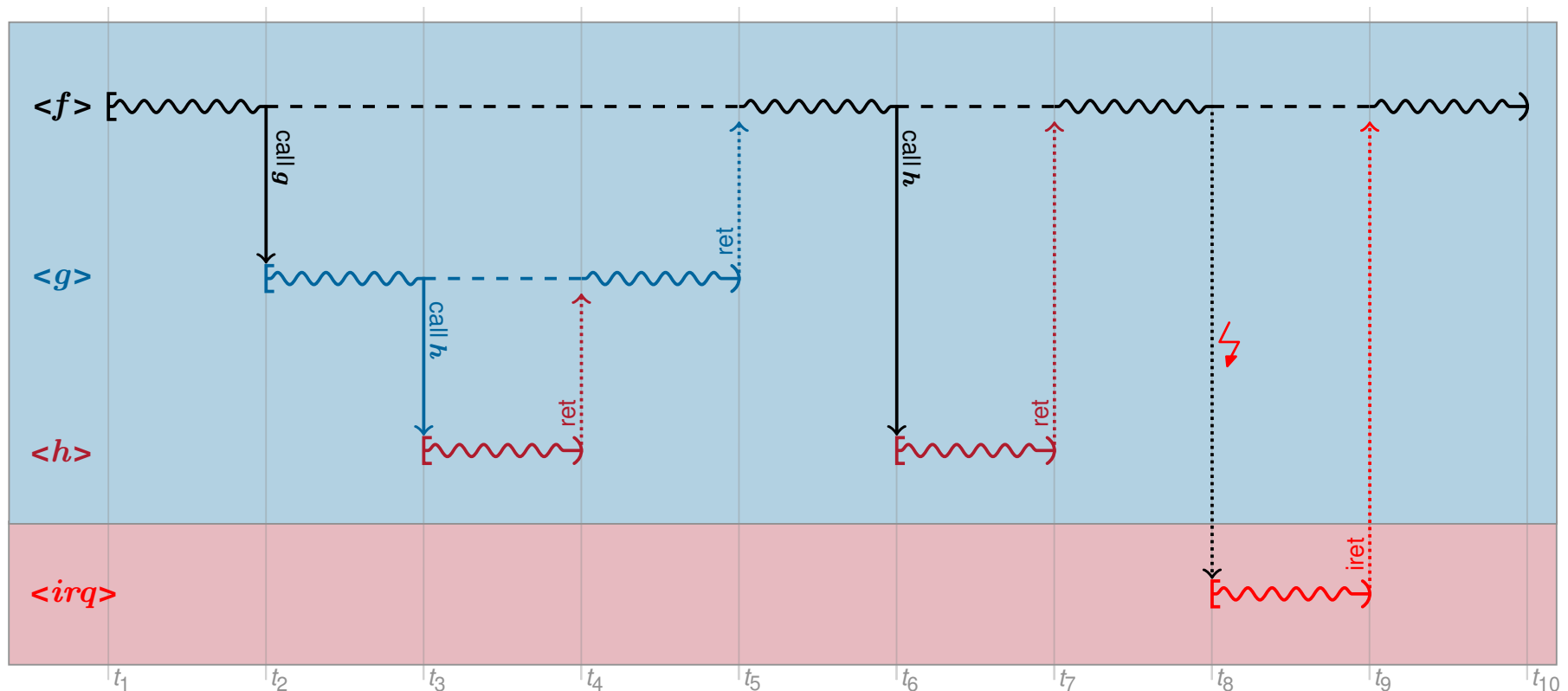
# Routinen $\mapsto$ asymmetrisches Fortsetzungsmodell

- Routinen-Kontrollflüsse bilden eine **Fortsetzungshierarchie**
  - Vater–Kind Relation zwischen Erzeuger und Erzeugtem
- Aktivierte Kontrollflüsse werden nach **LIFO** fortgesetzt
  - Der zuletzt aktivierte Kontrollfluss terminiert immer zuerst
  - Vater wird erst fortgesetzt, wenn Kind terminiert



# Routinen $\mapsto$ asymmetrisches Fortsetzungsmodell

- Das gilt auch bei **Unterbrechungen**
  - $\langle f \rangle \xrightarrow{\text{irq}}$  ist wie *call*, nur implizit
- Unterbrechungen können als **implizit** erzeugte und aktivierte Routinen-Ausführungen verstanden werden



# Grundbegriffe: Koroutine

- **Koroutine** (engl. *Coroutine*): verallgemeinerte Routine [1]
  - erlaubt zusätzlich: expliziten Austritt und Wiedereintritt
  - Sprachmittel *einiger* Programmiersprachen
    - z. B. Modula-2, Simula-67, Stackless Python
  - wird ausgeführt durch **Koroutinen-Kontrollfluss**
- **Koroutinen-Kontrollfluss**: eine Koroutine in Ausführung
  - Kontrollfluss mit eigenem, unabhängigen Zustand
    - mindestens Programmzähler (PC)
    - zusätzlich je nach (zu unterstützendem) **Compiler / ABI / ISA**: weitere Register, Stapel, ...
    - Im Prinzip ein eigenständiger Faden (engl. *Thread*) – **dazu später mehr**

**Koroutinen** und **Koroutinen-Kontrollflüsse** stehen ebenfalls in einer **Schema–Instanz Relation**.

In der Literatur ist diese Unterscheidung unüblich  $\rightsquigarrow$  Koroutinen-Kontrollflüsse werden (vereinfacht) ebenfalls als Koroutinen bezeichnet.



# Grundbegriffe: Koroutine

- Koroutinen-Kontrollflüsse werden erzeugt, gesteuert, und zerstört über zusätzliche **Elementaroperationen**
  - *create*  $g$ 
    1. **erzeugt** neue Koroutinen-Ausführung  $\langle g \rangle$  von  $g$
  - $\langle f \rangle$  *resume*  $\langle g \rangle$ 
    1. **suspendiert** die Koroutinen-Ausführung  $\langle f \rangle$
    2. **(re-)aktiviert** die Koroutinen-Ausführung  $\langle g \rangle$
  - *destroy*  $\langle g \rangle$ 
    1. **zerstört** die Koroutinen-Ausführung  $\langle g \rangle$

**Unterschied** zu Routinen-Kontrollflüssen: [SP, C 10-8]

Aktivierung und Reaktivierung sind  
**zeitlich entkoppelt** von Erzeugung und Zerstörung.

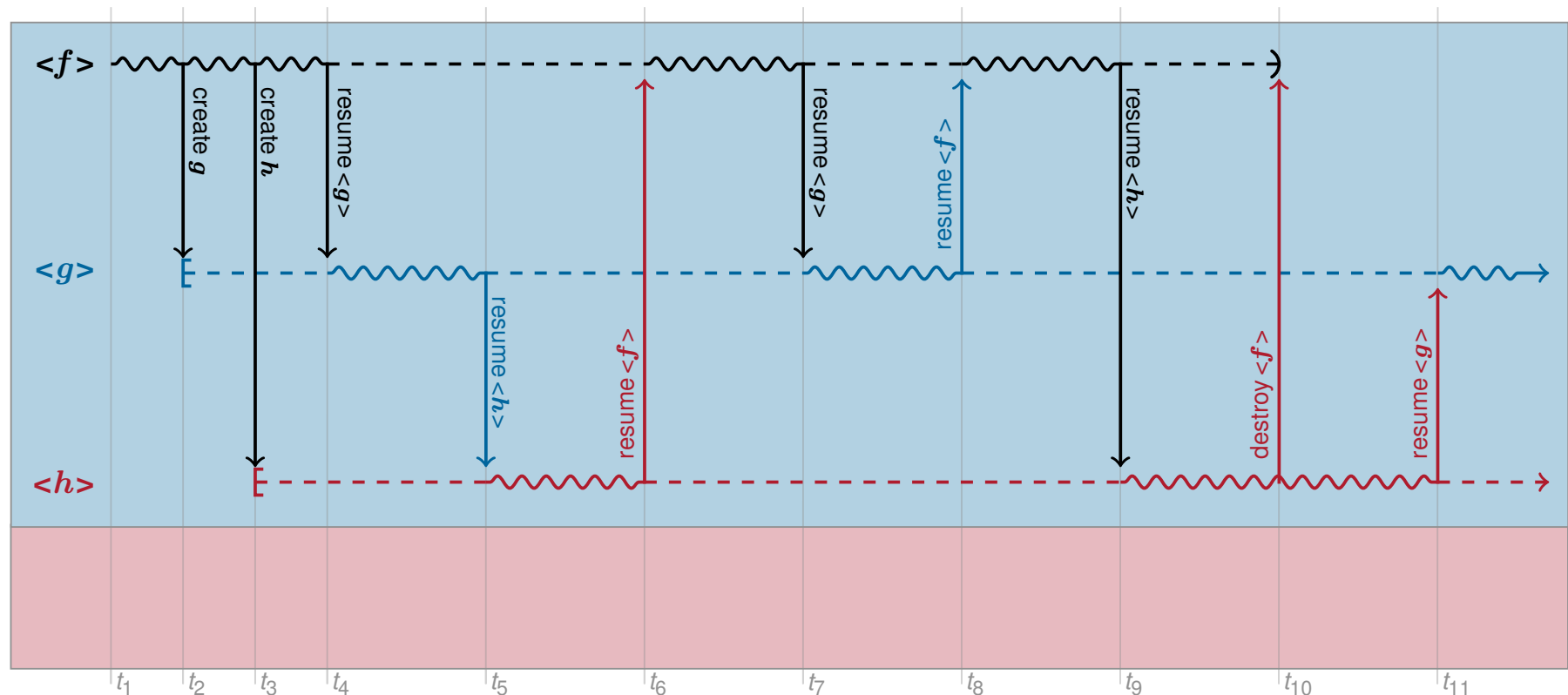
~> Koroutinen sind **echt mächtiger** als Routinen.





# Koroutinen $\mapsto$ symmetrisches Fortsetzungsmodell

- Koroutinen-Kontrollflüsse bilden eine **Fortsetzungsfolge**
  - Koroutinenzustand bleibt über Ein-/Austritte hinweg erhalten
- Alle Koroutinen-Kontrollflüsse sind **gleichberechtigt**
  - kooperatives Multitasking
  - Fortsetzungsreihenfolge ist beliebig



# Koroutinen und Programmfäden

---

- Koroutinen-Kontrollflüsse werden oft auch bezeichnet als
  - kooperative **Fäden** (engl. *cooperative Threads*)
  - **Fasern** (engl. *Fibers*)
- Das ist im Prinzip richtig, die Begriffe entstammen jedoch aus verschiedenen Welten
  - Koroutinen-Unterstützung ist historisch (eher) ein **Sprach**merkmal
  - Mehrfädigkeit ist historisch (eher) ein **Betriebssystem**merkmal
  - Die Grenzen sind fließend
    - *Sprachfunktion* — (*Laufzeit-*)*Bibliothekfunktion* — *Betriebssystemfunktion*
- Wir verstehen Koroutinen als **technisches** Konzept
  - um Mehrfädigkeit im BS zu implementieren
  - insbesondere später auch nicht-kooperative Fäden

